

Express.js Tham Khảo Nhanh

Routing, middleware, request, response, các pattern

Thiết Lập

Tạo & Khởi Động Server

```
const express = require("express");
const app = express();
app.listen(3000, () => console.log("Đang chạy trên :3000"));
```

Middleware Tích Hợp

```
app.use(express.json()); // parse body JSON
app.use(express.urlencoded({ extended: true })); // form data
app.use(express.static("public")); // phục vụ file tĩnh
```

Routing

HTTP Methods

```
app.get("/users", (req, res) => res.json(users));
app.post("/users", (req, res) => res.status(201).json(req.body));
app.put("/users/:id", (req, res) => res.json(updated));
app.delete("/users/:id", (req, res) => res.sendStatus(204));
```

Tham Số Route

```
app.get("/users/:id", (req, res) => {
  const { id } = req.params;
  res.json({ id });
});
```

Query String

```
// GET /search?q=express&page=2
app.get("/search", (req, res) => {
  const { q, page } = req.query;
  res.json({ q, page });
});
```

Middleware

Cấp Ứng Dụng

```
app.use((req, res, next) => {
  console.log(`${req.method} ${req.url}`);
  next();
});
```

Cấp Route

```
const auth = (req, res, next) => {
  if (!req.headers.authorization) return res.sendStatus(401);
  next();
};
app.get("/secret", auth, (req, res) => res.json({ ok: true }));
```

Thứ Tự Thực Thi

app.use(fn)	Chạy cho mọi request (theo thứ tự)
app.use(path, fn)	Chỉ chạy khi path prefix khớp
next()	Chuyển quyền điều khiển cho middleware tiếp theo
next(err)	Bỏ qua đến error handler

Request & Response

Request Object

req.params	Tham số route (/users/:id)
req.query	Query string (?key=val)
req.body	Body request đã parse (cần parser)
req.headers	Object chứa request header
req.method	HTTP method (GET, POST, ...)
req.path	Đường dẫn URL
req.cookies	Cookie (cần cookie-parser)

Response Object

res.json(obj)	Gửi response JSON
res.send(body)	Gửi string/Buffer/object
res.status(code)	Đặt HTTP status (có thể chain)
res.redirect(url)	Redirect 302 (hoặc truyền status)
res.sendFile(path)	Gửi file làm response
res.sendStatus(code)	Gửi status với text mặc định
res.set(header, val)	Đặt response header

Xử Lý Lỗi

Error Middleware

```
// Phải có 4 tham số - Express nhận ra là error handler
app.use((err, req, res, next) => {
  console.error(err.stack);
  res.status(err.status || 500).json({ error: err.message });
});
```

Định nghĩa error handler sau tất cả app.use() và route

Async Errors

```
// Bọc async route để bắt rejection
const wrap = (fn) => (req, res, next) =>
  Promise.resolve(fn(req, res, next)).catch(next);

app.get("/data", wrap(async (req, res) => {
  const data = await fetchData();
  res.json(data);
})));
```

Static Files

Phục Vụ Thư Mục Tĩnh

```
app.use(express.static("public"));
// phục vụ public/style.css tại /style.css

// Với virtual path prefix
app.use("/assets", express.static("public"));
// phục vụ public/style.css tại /assets/style.css
```

Tùy Chọn

dotfiles	'ignore' 'allow' 'deny'
maxAge	Cache-Control max-age tính bằng ms
index	Tên file index (mặc định: index.html)
fallthrough	Chuyển sang middleware tiếp theo khi 404

Templates

Thiết Lập View Engine

```
app.set("view engine", "ejs");
app.set("views", "./views");

app.get("/", (req, res) => {
  res.render("index", { title: "Trang chủ", items: [1, 2, 3] });
});
```

Engine Phổ Biến

ejs	Template JS nhúng (<%= val %>)
pug	Dựa trên thụt đầu dòng (trước đây là Jade)
handlebars	Kiểu Mustache ({{val}})

Router

Route Dạng Module

```
// routes/users.js
const router = require("express").Router();
router.get("/", (req, res) => res.json(users));
router.get("/:id", (req, res) => res.json(user));
module.exports = router;
```

Gắn Router

```
const usersRouter = require("./routes/users");
app.use("/api/users", usersRouter);
// GET /api/users -> router's "/"
// GET /api/users/5 -> router's "/:id"
```

Các Method Router

router.get/post/put/delete	Handler HTTP method
router.use(fn)	Middleware cấp router
router.param(name, fn)	Tiền xử lý tham số route
router.route(path)	Chain method trên một path

Pattern Xác Thực

JWT Middleware

```
const jwt = require("jsonwebtoken");
const auth = (req, res, next) => {
  const token = req.headers.authorization?.split(" ")[1];
  if (!token) return res.sendStatus(401);
  req.user = jwt.verify(token, process.env.SECRET);
  next();
};
```

Route Được Bảo Vệ

```
app.get("/profile", auth, (req, res) => {
  res.json({ user: req.user });
});
app.use("/api/admin", auth, adminRouter);
```

Các Pattern Thường Gặp

CORS

```
const cors = require("cors");
app.use(cors()); // cho phép mọi origin
app.use(cors({ origin: "https://example.com" })); // giới hạn
```

Môi Trường & Config

```
const port = process.env.PORT || 3000;
app.listen(port);

// Truy cập env trong route
if (app.get("env") === "production") {
  app.use(helmet());
}
```

Package npm Hữu Ích

cors	Cross-origin resource sharing
helmet	Security header
morgan	HTTP request logger
cookie-parser	Parse Cookie header
dotenv	Tải .env vào process.env
multer	Form data nhiều phần (upload file)